Name: Julianne Tillmann

Date: November 27, 2024

Course: IT FDN 110 A Au 24: Foundations Of Programming: Python

Assignment 07

Creating a Python Script with Classes, Class Inheritance, and Properties

Introduction

In Module 07, we explored how to use classes and objects to create and manage structured data effectively in Python. The assignment focuses on extending a previous course registration program by incorporating data classes, class inheritance, and properties. Additionally, the program emphasizes structured error handling to ensure robust functionality and user-friendly interactions.

This assignment required creating a Python program that registers students for courses, displays current enrollments, and saves data to a file using JSON format.

In creating this script, I worked from the provided starter file and made the necessary changes to meet the requirements. In addition, the latest version of the script is available on GitHub at DrJT2013/IntrotoProg-Python-Mod07

Creating the Script and Adding the Header

As with all assignments, I began by updating the script header with my name, date, and a brief description of the changes made. This ensures proper documentation and version control.

Creating the Main Body of the Script

Defining Constants and Variables

The script defines the following constants and variables:

- MENU: Displays the menu options to the user.
- FILE_NAME: Holds the name of the JSON file used for data storage.
- students: A list to hold all student records.

Classes and Functions

The program includes the following key classes and functions:

1. Person Class: Represents a base class for individuals with first and last names.
    - Attributes: first_name, last_name represent a person's first and last names
    - Methods:

        - __init__(self, first_name: str, last_name: str): Initializes first_name and last_name with validation
        - __str__(self): Returns the full name of the person
2. Student Class: Inherits from Person and adds course-specific details to represent a student enrolled in a course.
    - Attributes: Inherits first_name, last_name from Person, and adds course_name.
    - Methods:
        - __init__(self, first_name: str, last_name: str, course_name: str): Initializes attributes and calls the Person constructor
        - __str__(self): Returns the student's full name along with their course
3. FileProcessor Class: Handles reading and writing data to JSON files.
    - Methods:
        - read_data_from_file: Reads data from a JSON file into a list of student objects (i.e.,populates the student_data list).
        - write_data_to_file: Writes the list of student objects to a JSON file.
4. IO Class: Manages user input and output.
    - Methods:
        - output_menu: Displays menu options.
        - input_menu_choice: Gets a menu choice from the user.
        - input_student_data: Prompts the user to enter student information.
        - output_student_and_course_names: Displays all registered students and their courses.
        - output_error_messages: Displays error messages to the user.

Main Features of the Program

The program offers the following features:

- Menu Option 1: Registers a new student by prompting the user for their first name, last name, and course name.
- Menu Option 2: Displays all registered students and their courses.
- Menu Option 3: Saves the student data to a JSON file.
- Menu Option 4: Exits the program.

Each menu option is implemented with appropriate error handling to ensure the program remains functional in case of errors.

Testing the Script

The program was tested under the following scenarios:

- Starting Without a File: Successfully created a new file and notified the user (Figure 1).

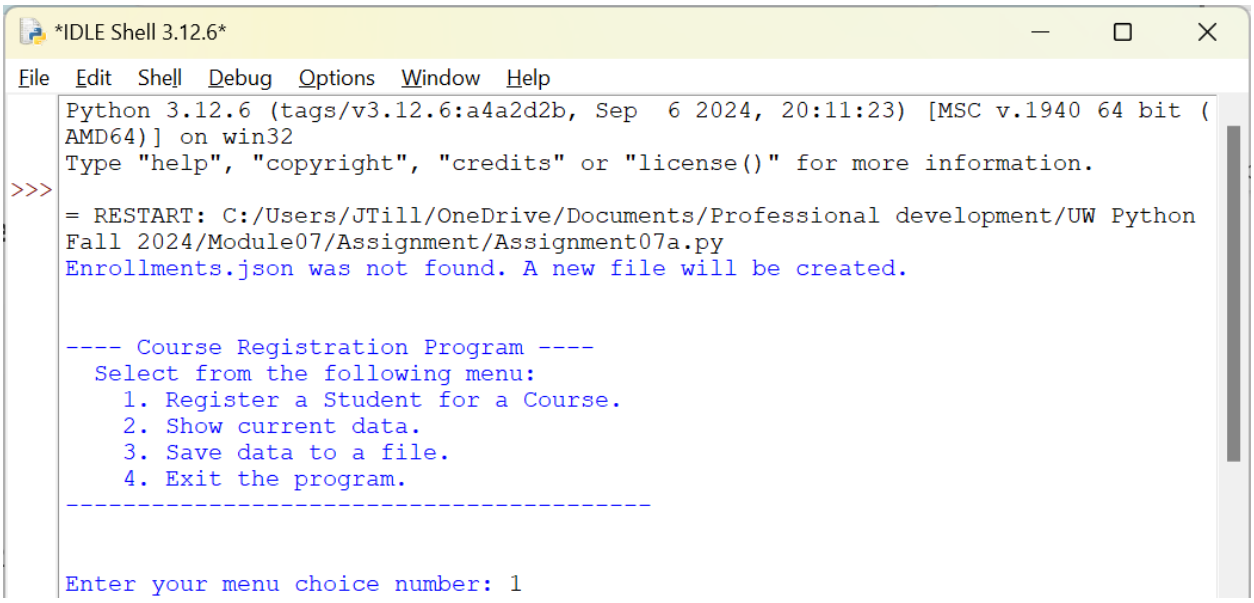

Figure 1. Running the script without a pre-existing file. A new file was created.

- Adding New Students: Registered students with valid inputs and rejected invalid inputs (Figures 2 and 3).

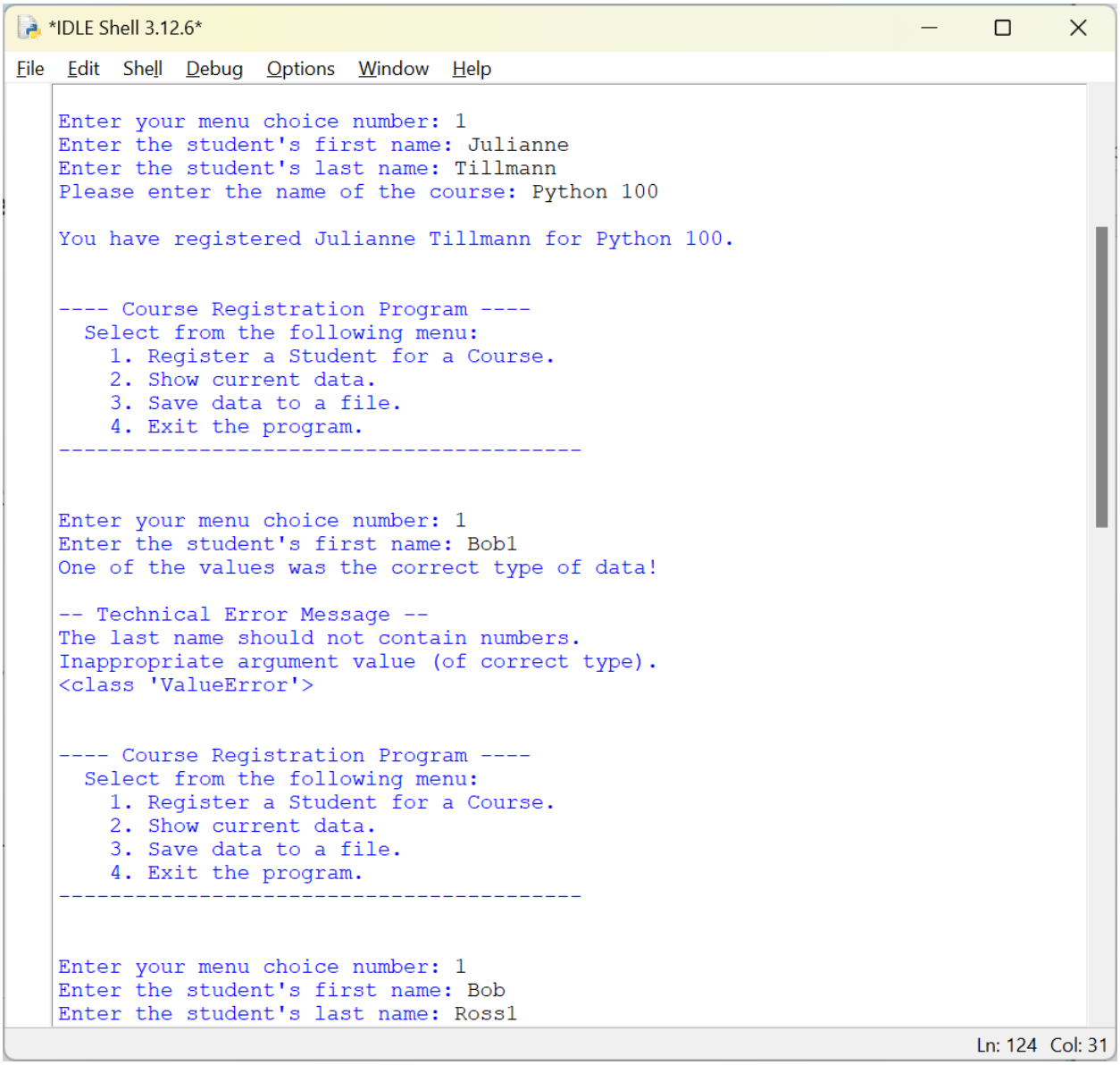

Figure 2. Registering new students.

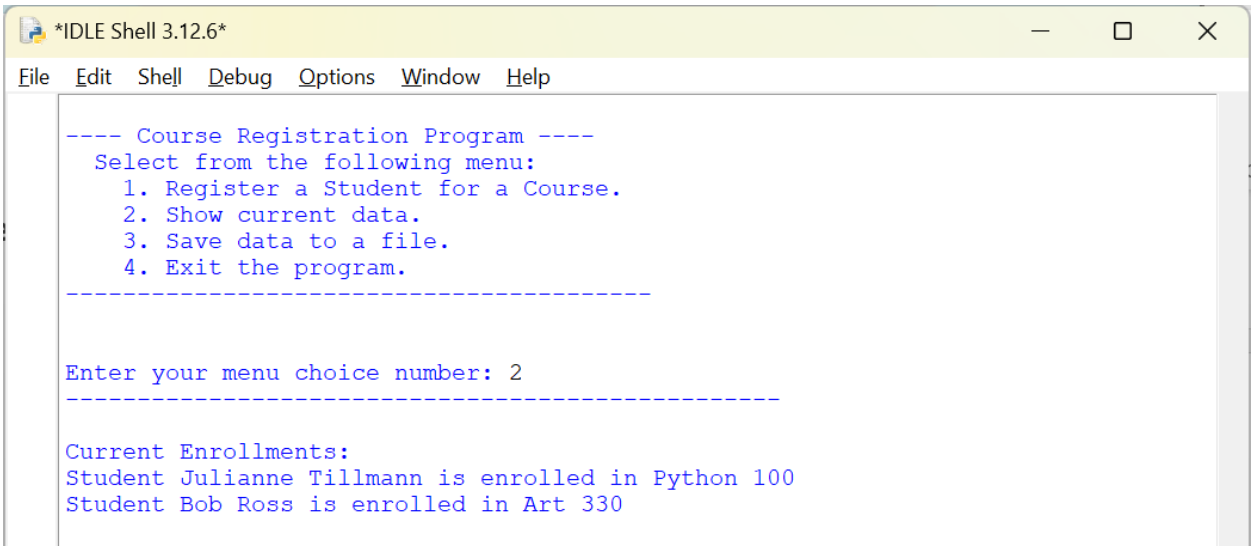

Figure 3. Reviewing the data with new students added.

- Reloading with Pre-Existing Data: Loaded data from the JSON file and displayed it accurately (see Figure 4).

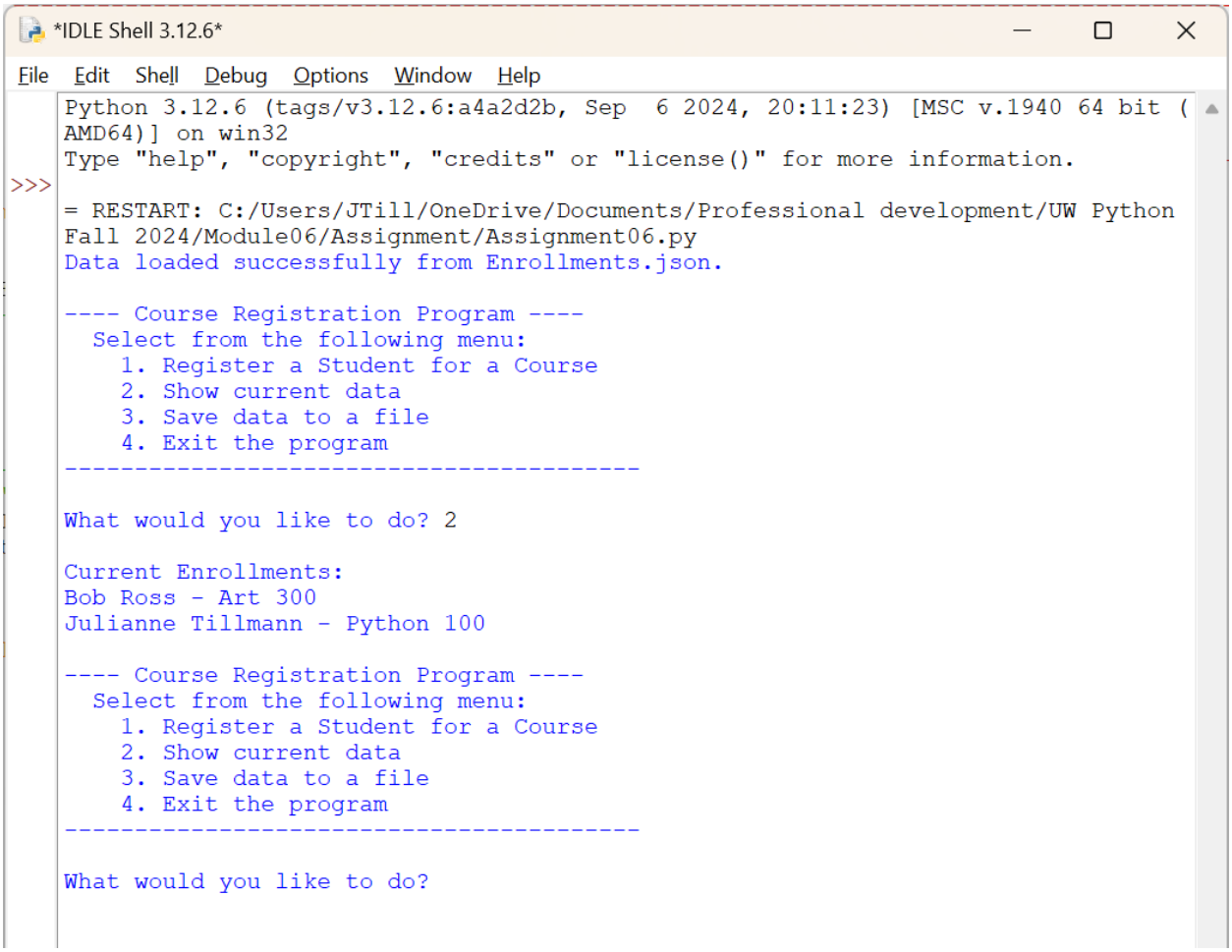

Figure 4. Ran the script with a pre-existing file and checked the data.

- Handling corrupt data files: Showed a message that there was a problem with the file (Figure 5).

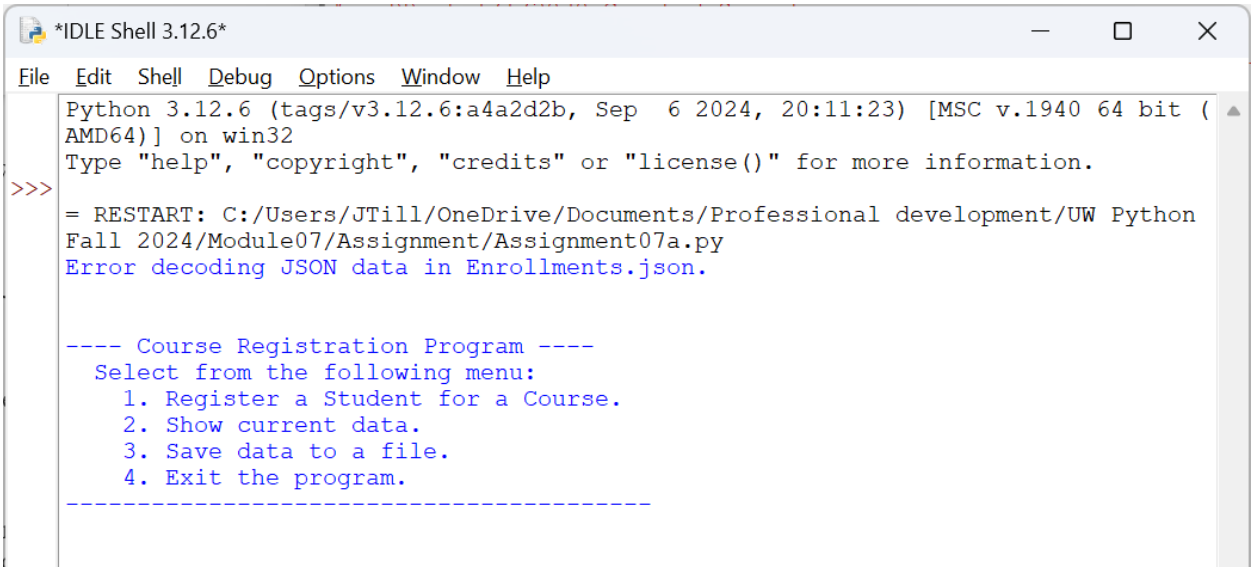

Figure 5. Problem loading a file with missing data.

The program's robust error handling ensured that all scenarios were handled gracefully, and data integrity was maintained.

Summary

This assignment enhanced my understanding of Python functions, classes, and error handling. By integrating these features into the course registration program, I improved its functionality and user experience. Testing the program thoroughly allowed me to identify and fix issues, ensuring the program performs reliably in various scenarios.